Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de la Universidad

Algoritmos y Estructuras de Datos

Tarea 1

Pilas de Arena

Nombre: Sebastián Jorquera
Profesor: Nelson Baloian
Auxiliares: Sergio Peñafiel
Gabriel Azocar
Ayudantes: Cristóbal Muñoz
Fabian González
Gabriel Chandia
Fecha de realización: 14 de abril de 2017
Fecha de entrega: 14 de abril de 2017
Santiago, Chile

1. Introducción

El fin último de estudiar diversos problemas físicos es encontrar un modelo analítico que permita describir el comportamiento de estos. Sin embargo muchas veces estos modelos son tan complejos que se vuelven casi imposibles de resolver mediante métodos de cálculo tradicionales; y es allí donde la modelación computacional ha aparecido como una alternativa para la resolución de estos problemas.

En nuestro caso, el problema a estudiar corresponde al comportamiento de los granos de arena apilados en un espacio, a los cuales se les aplica una regla de estabilidad por gravedad. En particular, el interés esta en observar el estado de evolución final de esta pila de granos arena.

El comportamiento de la pila de arena es relativamente simple. La superficie donde se ubican los granos de arena se separará en celdas cuadradas idénticas, formando una estructura con la siguiente forma:

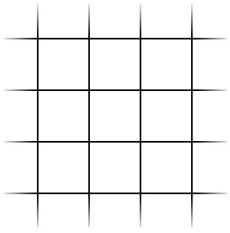


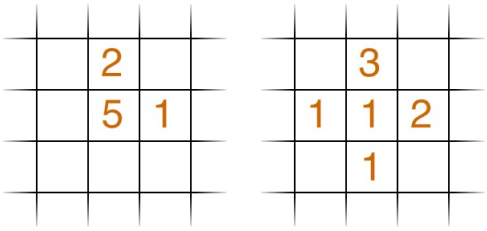
Figura 1.1: modelo de superficie

De esta forma, los granos de arena se encontrarán apilados dentro de estas celdas. La regla de estabilidad consiste en que si el número de granos dentro de una celda es mayor o igual a 4, la pila debe desbordarse y dejar un grano en cada celda adyacente.

Para representar la evolución de la pila de arena, se creará una matriz de un tamaño los suficientemente grande como para almacenar la configuración final de los granos de arena, con el total de granos de arena en la celda central en un inicio. Esta matriz será recorrida un número de veces tal que en cada pasada se aplique la regla de estabilidad a las cuadrillas con arena hasta llegar al punto en que no hayan más celdas inestables, momento en el cual se mostrará la distribución final de los granos de arena.

2. Análisis del problema

El problema a resolver es el siguiente; el usuario debe declarar un número N de granos de arena, que corresponde a la cantidad de granos que se ubicarán inicialmente en el centro. Luego de esto, se aplicará la siguiente regla de estabilidad mencionada en la sección anterior: si $n \geq 4$, con n el número de granos en una celda, entonces esta celda se desbordará, depositando un grano de arena en cada una de las celdas adyacentes. El comportamiento se muestra en la siguiente imagen.



(a) estado inicial de las cuadrillas

(b) estado posterior al desborde

Figura 2.1: Evolución de la pila de arena

2.1. Supuestos

El programa implementado para resolver el problema funciona bajo los siguientes supuestos.

- El número ingresado por el usuario siempre corresponde a un int
- La matriz en la que se distribuye la arena es lo suficientemente grande para almacenar la distribución final de los granos de arena.
- Dado el supuesto anterior, toda celda con arena tiene una celda adyacente arriba, abajo, a su izquierda y a su derecha (no existen sumideros).
- El orden en que se desbordan las celdas no tiene efecto en la configuración final.

2.2. Metodología

A continuación se muestran los pasos a seguir por el algoritmo para determinar la distribución final de los granos de arena.

1. Se le solicita al usuario un número inicial de granos de arena
2. Se crea una matriz que contiene solamente 0 lo suficientemente grande para almacenar la distribución final de arena (Como hacer esto se detallará en la siguiente sección)
3. Se cambia el valor de la celda central de la matriz por el número de granos de arena.
4. Se recorre la matriz, aplicando a cada celda la regla de estabilidad.
5. Se cuenta el número de desbordes o avalanchas ocurridas en un paso por la matriz.

6. Si el número de avalanchas es mayor a 0, se repiten los paso 4 y 5 hasta lograr recorrer la matriz sin que ocurran avalanchas.

7. se muestra la distribución final.

2.3. Casos de borde

- Si el número ingresado por el usuario no corresponde a un int, el programa no se ejecutará.

3. Solución del problema

La resolución del problema consta de dos pasos fundamentales:

1. Crear una matriz lo suficientemente grande para contener la distribución final de los granos de arena.

2. Ubicar el total de granos en el centro y recorrer la matriz, aplicando el criterio de estabilidad, hasta llegar a la configuración o distribución final.

A continuación se presentará la solución de cada uno de estos problemas por separado, y luego se mostrará la solución final.

3.1. Creación de la matriz

La idea es encontrar una cota superior para el tamaño de la matriz, y así poder almacenar toda la arena en su interior (evitar sumideros). Cuando se utiliza un número bajo de granos de arena (por ejemplo 25) la distribución final de los granos de arena en la matriz tiene la siguiente forma: En

$$\begin{bmatrix} 0 & 0 & x & 0 & 0 \\ 0 & x & x & x & 0 \\ x & x & x & x & x \\ 0 & x & x & x & 0 \\ 0 & 0 & x & 0 & 0 \end{bmatrix}$$

Figura 3.1: ejemplo de distribución de arena

esta figura, las celdas marcadas con una x corresponden a celdas con arena. De esta representación es posible extraer ciertas conclusiones, las cuales son:

- La forma de diamante es la estructura básica o 'esqueleto' de las distribuciones.
- la fila y la columna central son las que poseen el mayor número de celdas con arena.

Dado esto se puede ver que basta determinar una cota superior para la cantidad de celdas con arena en la columna o fila central (dado que el problema es simétrico, ambas tienen el mismo tamaño). Asumiendo que la distribución tiene forma de diamante, su "área"puede calcularse como:

$$Area = \frac{d^2}{2} \tag{3.1}$$

donde d corresponde al largo de la diagonal, y en este caso el área corresponde al número de celdas contenidas. Si además se asume que cada celda contiene solamente un grano de arena, se tiene que el área del diamante es igual al total de granos de arena, al cual denotaremos por N. Reemplazando en la ecuación (3.1) se tiene que:

$$N = \frac{d^2}{2} \tag{3.2}$$

y despejando d se llega a:

$$d = \sqrt{2 * N} \tag{3.3}$$

Como d es la diagonal, basta con aproximarla al entero mayor más cercano (pues el resultado casi nunca es entero). De esta forma se determina que basta con una matriz de tamaño [d]X[d] para contener la arena. Así, se define un algoritmo que permite crear una matriz del tamaño deseado, dado una cantidad N de granos:

Lista de Códigos Fuente 1: Algoritmo generador de matrices

```
1  public static int[][] matrix_generator(int N){
2      double size = Math.ceil(Math.sqrt(N));
3      if(size%2!=0) {
4          int[][] matriz = new int[(int)size][(int)size];
5          return matriz;
6      }
7      else{
8          size+=1;
9          int[][] matriz = new int[(int)size][(int)size];
10         return matriz;
11     }
12  }
```

El algoritmo sigue la idea descrita anteriormente para determinar el tamaño de la matriz, con unas pequeñas modificaciones. En primer lugar, luego de probar una versión inicial del programa, se pudo ver que bastaba con calcular solamente $\sqrt{N}$ en lugar de $\sqrt{2N}$.Por último, si el valor del tamaño es un número par, se aumenta en uno para obtener un valor impar; esto para siempre tener una celda central bien definida.

3.2. Aplicar criterio de estabilidad

Una vez creada la matriz, es necesario ubicar la matriz y aplicar el criterio de estabilidad hasta llegar a la distribución final. Para esto se implementa el siguiente algoritmo:

Lista de Códigos Fuente 2: Algoritmo repasador de matriz

```
1   public static int avalanche_matrix(int[][] matriz) {
2       int avalanchas = 0;
3       for (int i = 0; i < matriz.length; i++) {
4           for (int j = 0; j < matriz.length; j++) {
5               if (matriz[i][j] >= 4) {
6                   matriz[i + 1][j] += 1;
7                   matriz[i - 1][j] += 1;
8                   matriz[i][j + 1] += 1;
9                   matriz[i][j - 1] += 1;
10                  matriz[i][j] -= 4;
11                  avalanchas += 1;
12              }
13          }
14      }
15      return avalanchas;
16  }
```

El algoritmo recorre toda la matriz, y al encontrar una celda que no cumpla el criterio de estabilidad descrito anteriormente, genera un desborde y depositan un grano de arena en cada celda adyacente. Además, se cuenta el número de avalanchas totales y se retorna este valor.

El algoritmo *avalanche_matrix* solamente recorre la matriz una vez. Para poder aplicarlo la cantidad de veces necesaria, se aplica la siguiente instrucción.

Lista de Códigos Fuente 3: Loop del recorrido de la matriz

```
1  public static void main(String[] args){
2      //...
3      while(true){
4          int av = avalanche_matrix(matriz);
5          if(av==0){
6              break;
7          }
8      }
9      //...
10 }
```

Este loop permite ejecutar *avalanche_matrix* hasta llegar a la configuración de la matriz donde ya no se tienen más avalanchas.

Finalmente, el módulo PilaArena tiene la siguiente forma.

Lista de Códigos Fuente 4: Loop del recorrido de la matriz

```
1  public static void main(String[] args){
2      Scanner granos = new Scanner(System.in);
3      System.out.println("Numero de granos (N)?");
4      int N = granos.nextInt();
5      int[][] matriz = matrix_generator(N);
6      int centro = (int) Math.floor(matriz.length/2);
7      matriz[centro][centro] = N;
8      while(true){
9          int av = avalanche_matrix(matriz);
10         if(av==0){
11             break;
12         }
13     }
14     Ventana v = new Ventana(800, "resultado");
15     v.mostrarMatriz(matriz);
16 }
```

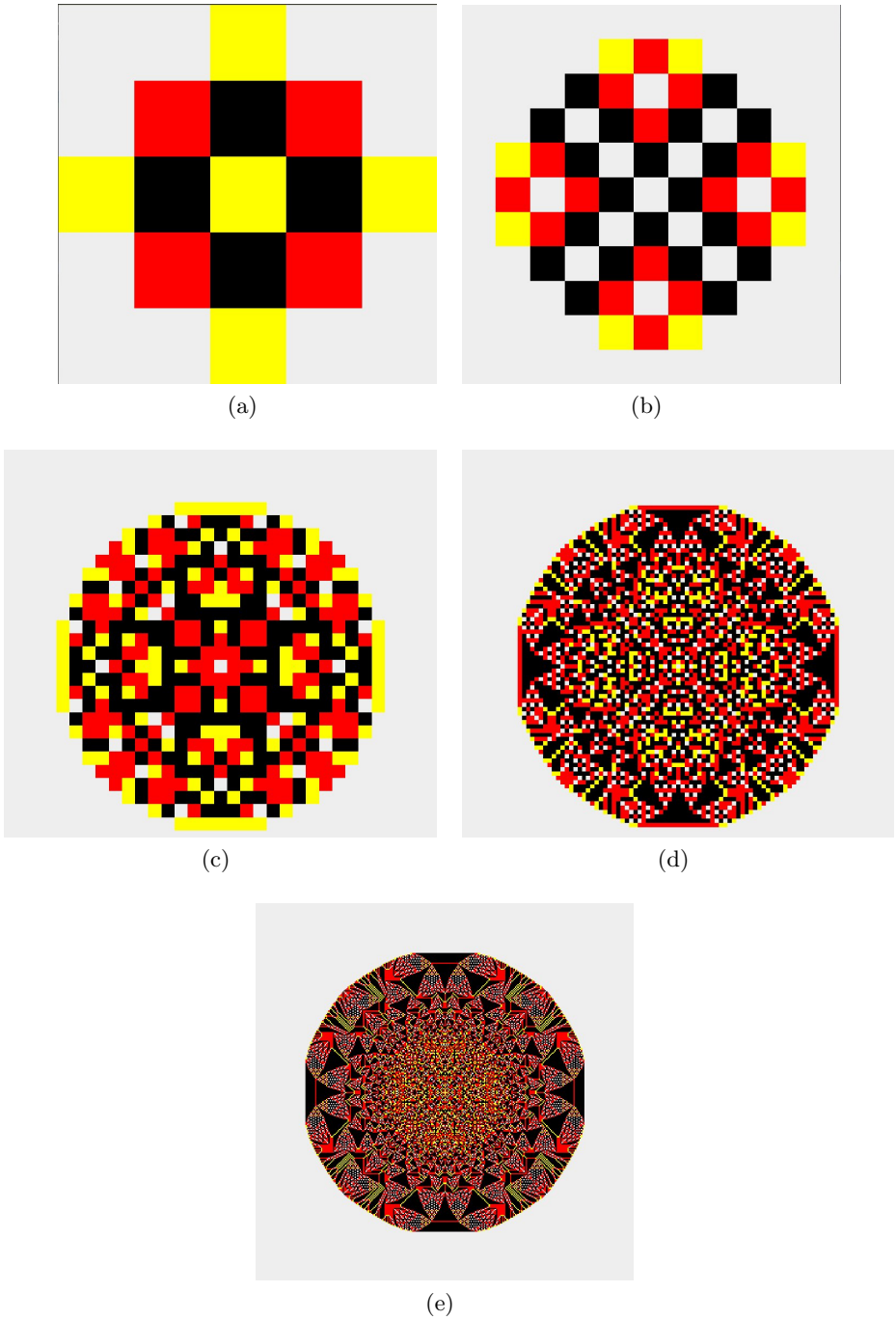
3.3. Modo de uso

Todos los métodos necesario para resolver el problema están contenidos en el archivo *PilaArena.java*. Una vez compilado, basta con llamar a PilaArena.

Los métodos necesarios para mostrar las imágenes se encuentran en el archivo *Ventana.java*, el cual se entrego al inicio de la tarea.

4. Resultados y Análisis

El programa se ejecutó cinco veces, con 25, 100, 1000, 10000, 100000 granos de arena respectivamente. Se presentan las configuraciones finales obtenidas.



(a) (b) (c) (d) (e)

Figura 4.1: Distribución final con 25, 100, 1000, 10000, 100000 granos de arena

4.1. Análisis de Resultados

Si bien no se tenía conocimiento previo de como debían ser las distribuciones finales para las cantidades de granos de arena utilizados, hay indicadores que permiten concluir que estas están bien construidas. En primer lugar se observa que las imágenes son simétricas, lo que es acorde a la forma de evolución de la pila de arena. Además en ningún caso se obtuvo un error con respecto al número de granos en las celdas (ninguna celda tiene un valor mayor a 4)

4.2. Análisis de Algoritmos

Dentro de los métodos implementados, existen 2 que merecen ser analizados con más detalle. Estos son el método *avalanche_matrix* y el loop booleano en dentro del *main*.

Ambos elementos están sumamente relacionados, puesto que como se dijo antes el loop booleano es el que permite ejecutar el método *avalanche_matrix* la cantidad necesaria de veces para obtener el resultado final. Implementar un loop booleano conlleva el riesgo de que el programa quede atrápado en un ciclo de repeticiones infinito. Este loop podría evitarse si el método *avalanche_matrix* se hiciera recursivo; de hecho este fue implementado de esa manera en un principio. El problema es que al ser recursivo, existía un valor límite para los granos de arena; al sobrepasar este número, se producía un *StackOverflow*, y el programa no se ejecutaba. Dado esto se optó por el loop booleano, pero no se descarta la posibilidad de encontrar un método recursivo para obtener los resultados solicitados.

5. Anexos

Se muestra el programa completo utilizado para modelar las pilas de arena.

Lista de Códigos Fuente 5: Loop del recorrido de la matriz

```
import java.util.Scanner;

/**
 * Created by sebastian on 07-04-17.
 */
public class PilaArena {

    public static int[][] matrix_generator(int N){
        // se calcula la cota superior para el tamao de la fila central
        double size = Math.ceil(Math.sqrt(N));
        if(size%2!=0) {
            // si el valor es impar, se deja como el tamao de la matriz
            int[][] matriz = new int[(int)size][(int)size];
            return matriz;
        }
        else{
            // si el valor es par, se le suma 1 para hacerlo impar y se crea la matriz
            size+=1;
            int[][] matriz = new int[(int)size][(int)size];
            return matriz;
        }
    }



    public static int avalanche_matrix(int[][] matriz) {
        // se inicializa un contador para las avlanchas
        int avalanchas = 0;
        for (int i = 0; i < matriz.length; i++) {
            // se recorren todas las filas de la matriz
            for (int j = 0; j < matriz.length; j++) {
                // se recorren todas las celdas de la fila
                if (matriz[i][j] >= 4) {
                    // se aplica el criterio de estabilidad y se suma 1 al numero de
                        avalanchas
                    matriz[i + 1][j] += 1;
                    matriz[i - 1][j] += 1;
                    matriz[i][j + 1] += 1;
                    matriz[i][j - 1] += 1;
                    matriz[i][j] -= 4;
                    avalanchas += 1;
                }
            }
        }
        // se devuelve el numero de avalanchas
        return avalanchas;
    }
```

```
47
48      public static void main(String[] args){
49          Scanner granos = new Scanner(System.in);
50          // se solicita el numero de granos al ususario
51          System.out.println("Numero de granos (N)?");
52          int N = granos.nextInt();
53          // se genera la matriz con el tamano necesario
54          int[][] matriz = matrix_generator(N);
55          // se depositan los granos de arena en el centro
56          int centro = (int) Math.floor(matriz.length/2);
57          matriz[centro][centro] = N;
58          // se recorre la matriz hasta que ya no se produzcan mas avalanchas
59          while(true){
60              int av = avalanche_matrix(matriz);
61              if(av==0){
62                  break;
63              }
64          }
65          // finalmente, se muestra la distribucion final de los granos de arena
66          Ventana v = new Ventana(800, "resultado");
67          v.mostrarMatriz(matriz);
68      }
69  }
```